INSTITUTIONEN FÖR
NATURVETENSKAP OCH TEKNIK

Introduktion till ingenjörsarbete inom datateknik, DT502G

Grundläggande programmering, teori
1,5 högskolepoäng

Skriftlig tentamen

2019-11-19

Introduktion till ingenjörsarbete inom datateknik, DT502G

Tillåtna hjälpmedel: penna, radergummi.

Instruktioner:

- **Läs igenom alla frågor noga.**
- **Skriv bara på ena sidan av svarsbladet.**
- **Skriv tentamenskoden på varje svarsblad.**
- **Du kan svara på *Svenska* eller *Engelska*.**
- ***Skriv läsligt!***

Ansvarig lärare: Andreas Persson, tel: 019-303714 (alt. 0707760861), och Amy Loutfi, tel: 0722149853

För betyg G krävs 50% av total poäng.

Lycka till!

Part I: Open answers questions

Answer each question on a separate page.

Question 1

What will be the print output of the script below? *Justify your answer.* **(2p)**

```
def selector(x):
    if x > 1 and x < 4:
        print("A", end=" ")  # Note, "end" parameter is here used to
                             # ...override default newline.
    elif x == 2:
        print("B", end=" ")

    elif x > 2 and x <= 4:
        print("C", end=" ")

    else:
        print("D", end=" ")

if __name__ == "__main__":
    for i in range(5):
        selector(i)
```

Question 2

Write a Python function that uses *Caesar cipher* to *encrypt* a text string by *shifting* each letter in the string some fixed number of positions up/down the alphabet. **(3p)**

Below is an example of the desired output of your function:

```
>>> txt = "pythons are snakes"
>>> encrypted = caesarCipher(txt, 1)
>>> print(encrypted)
qzuipot!bsf!toblft
...
>>> decrypted = caesarCipher(encrypted, -1)
>>> print(decrypted)
pythons are snakes
```

***Tip**, use built-in Python functions* `ord` *and* `chr` *for translation between characters and Unicode codes (i.e., integer numbers), and vice versa.*

***Note**, you do not need to handle the "wrapping" of the alphabet (i.e., if a letter is shifted past the end of the alphabet, the letter wraps around to the beginning) in order to get full points.*

Question 3

Based on the following *library script*:

```
01 from time import time
02
03 class Timing:
04
05     def __init__(self):     # Question a)
06         self.t = time()
07
08     def reset(self):
09         self.t = time()
10
11     def stop(self):
12         self.t = time() - self.t
13
14     def log(self, msg):
15         print("{0}: {1:0.3f}".format(msg, self.t))
16
17 if __name__ == "__main__":  # Question b)
18     t = Timing()
19     even = []
20     t.reset()
21     for num in range(100000000):
22         if num % 2 == 0:
23             even.append(num*num)
24     t.stop()
25     t.log("Processing time")
```

a) Explain the use of the *special class method* on *line 5* in relation to the example of *lines 18-25.* **(1p)**

b) Explain the purpose of the *condition* on *line 17* in relation to *importing* the *library script* from another script. **(1p)**

Question 4

The function below uses nested **`for`** loops to *sort a list* in ascending order, e.g. for a list `[8, 3, 9, 5]`, the sorted result of this function would be: `[3, 5, 8, 9]`

```
def sort(x):
    for i in range(0, len(x)):
        for j in range(i, len(x)):
            if x[i] > x[j]:
                x[i], x[j] = x[j], x[i]
```

Your task is to rewrite the function above, but using **`while`** loops instead. **(2p)**

Question 5

The *mean* and *standard deviation* of a given dataset can be calculated according to the following formulas, respectively: $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$, $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \left|x_i - \mu\right|^2}$

Where N is the total *number of elements* in the dataset.

a) Your task is to write a function `mean(x)` that accepts an *arbitrary list of floats* as a parameter and uses a *loop structure* (of your chose) in order to calculate and *return* the *mean value* of the list (according to the first formula above). **(2p)**

b) Next, write a corresponding function `std(x)` that accepts the same *arbitrary list of floats* as a parameter and uses a *loop structure* (of your chose) together with your `mean` -function in order to calculate and *return* the *standard deviation* of the list (according to the second formula above). **(2p)**

***Tip**, the* `math` *-library contains an* `sqrt` *-function* for calculating the *square root.*

Part II: Multiple choice questions

Answer by selecting the option(s) that you think best matches the question.

It is recommended that you mark your answer directly in conjunction with each question and hand-in this page together with the following pages for this part.

Question 6

Both functions below are meant to be *fault-tolerant functions* for handle *conversion from a string to an integer*.

A:
```
def str2int(s):
    try:
        i = int(s)
        return i
    except ValueError as e:
        return 0
```

B:
```
def str2int(s):
    i = eval(s)
    if type(i) != int:
        i = round(i)
    return i
```

However, one (or both) of the functions above is not a solution that will correctly handle the following user input (with the purpose of solving the integer division `12 // 3 = 4`):

```
>>> x, y = input("Enter two integers: ").split(',')
Enter two integers: 12.01, 2.99
>>> x = str2int(x)
>>> y = str2int(y)
>>> print(x // y)
???
```

Complete the sentence below by filling in the blanks (based on given options): **(2p)**

Function A _____ , while function B _____.

a) will *correctly* print the *quotient* `4`

b) will *incorrectly* print the *quotient* `6`

c) will *incorrectly* print the *quotient* `0`

d) will *raise* a `ZeroDivisionError` exception and, therefore, print *nothing*

Question 7

A. Which of the following *data type* and *data collections* are *mutable* (meaning that an object of the data type/data collection is changeable)? *Note, more than one option might be correct!* **(1p)**

a) `str` (or string) **b)** `list` **c)** `dict` (or dictionary) **d)** `tuple`

B. For which of the following *data type* or *data collection* are the *elements* (or *characters*) **not** accessible through *indexing*? **(1p)**

a) `str` (or string) **b)** `list` **c)** `dict` (or dictionary) **d)** `tuple`

Question 8

Consider the following function:

```
def fatalStr(x, y):
    txt = "fatal system error"
    return txt.split(" ")[x][y] + txt.split(" ")[y][x]
```

What is the correct function call for printing out the letters `"rm"`? **(1p)**

a) `print(fatalStr(-1, 1))`
b) `print(fatalStr(1, -1))`
c) `print(fatalStr(0, 2))`
d) `print(fatalStr(-2, 0))`

Question 9

Analyze the following *boolean expression*:

```
(x and not y) or (not x and y)
```

What is the correct *truth table* for the expression above? **(1p)**

x	y	a)	b)	c)	d)
False	False	False	True	False	True
True	False	False	False	True	False
False	True	False	False	True	False
True	True	True	False	False	True

Question 10

The following function is meant to *recursively* calculate the *factorial* of a *non-negative integer number* (`n`). E.g., in case `n = 5`, the function should return: $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

```
def factorial(n):
    if _____________?
        return 1
    return n * factorial(n - 1)
```

What is the correct *base case* (i.e., termination criteria) for the function above? **(1p)**

a) `n > 1` **b)** `n >= 1` **c)** `n != 1` **d)** `n == 1`